Principles of AI Planning

5. Planning as search: progression and regression

EREI BURO

Albert-Ludwigs-Universität Freiburg

Bernhard Nebel and Robert Mattmüller

October 30th, 2013



Planning as (classical) search

Search

Introduction Classification

Progressio

Regression

What do we mean by search?



- Search is a very generic term.
- Every algorithm that tries out various alternatives can be said to "search" in some way.
 - Here, we mean classical search algorithms.
 - Search nodes are expanded to generate successor nodes.
 - Examples: breadth-first search, A*, hill-climbing, ...
 - To be brief, we just say search in the following (not "classical search").

Search

Classification

Dogranaian

Cummany

Do you know this stuff already?



- We assume prior knowledge of basic search algorithms:
 - uninformed vs. informed
 - systematic vs. local
- There will be a small refresher in the next chapter.
- Background: Russell & Norvig, Artificial Intelligence –
 A Modern Approach, Ch. 3 (all of it), Ch. 4 (local search)

Search Introduction

Classification

Regression

Search in planning



- search: one of the big success stories of Al
- many planning algorithms based on classical AI search (we'll see some other algorithms later, though)
- will be the focus of this and the following chapters (the majority of the course)

Search

Classification

Regression

C-----

Satisficing or optimal planning?



Must carefully distinguish two different problems:

- satisficing planning: any solution is OK (although shorter solutions typically preferred)
- optimal planning: plans must have shortest possible length

Introduction

Regression

Both are often solved by search, but:

- details are very different
- almost no overlap between good techniques for satisficing planning and good techniques for optimal planning
- many problems that are trivial for satisficing planners are impossibly hard for optimal planners



How to apply search to planning? → many choices to make!

Choice 1: Search direction

- progression: forward from initial state to goal
- regression: backward from goal states to initial state
- bidirectional search

Search

Classification

Regression



How to apply search to planning? → many choices to make!

Choice 2: Search space representation

- search nodes are associated with states (state-space search)
- search nodes are associated with sets of states

Introduction

Classification

Regression



How to apply search to planning? → many choices to make!

Choice 3: Search algorithm

- uninformed search: depth-first, breadth-first, iterative depth-first, ...
- heuristic search (systematic): greedy best-first, A*, Weighted A*, IDA*, ...
- heuristic search (local): hill-climbing, simulated annealing, beam search, ...

Search

Classification

Regression



How to apply search to planning? → many choices to make!

Choice 4: Search control

- heuristics for informed search algorithms
- pruning techniques: invariants, symmetry elimination, partial-order reduction, helpful actions pruning, ...

Classification

Search-based satisficing planners





FF (Hoffmann & Nebel, 2001)

- search direction: forward search
- search space representation: single states
- search algorithm: enforced hill-climbing (informed local)
- heuristic: FF heuristic (inadmissible)
- pruning technique: helpful actions (incomplete)

→ one of the best satisficing planners

Introduction Classification

.

Regression





Fast Downward Stone Soup (Helmert et al., 2011)

- search direction: forward search
- search space representation: single states
- search algorithm: A* (informed systematic)
- heuristic: multiple admissible heuristics combined into a heuristic portfolio (LM-cut, M&S, blind, ...)
- pruning technique: none

→ one of the best optimal planners

Introduction

_

Regression

Our plan for the next lectures



Choices to make:

- search direction: progression/regression/both
- search space representation: states/sets of states → this chapter
- search algorithm: uninformed/heuristic; systematic/local → next chapter
- search control: heuristics, pruning techniques → following chapters

Classification

Regression



Search

Progression

Over

Regression

Summary

Progression

Planning by forward search: progression



Progression: Computing the successor state $app_o(s)$ of a state s with respect to an operator o.

Progression planners find solutions by forward search:

- start from initial state
- iteratively pick a previously generated state and progress it through an operator, generating a new state
- solution found when a goal state generated

pro: very easy and efficient to implement

Searcl

Progression

Regression

Search space representation in progression planners





Two alternative search spaces for progression planners:

- search nodes correspond to states
 - when the same state is generated along different paths, it is not considered again (duplicate detection)
 - pro: save time to consider same state again
 - con: memory intensive (must maintain closed list)
- 2 search nodes correspond to operator sequences
 - different operator sequences may lead to identical states (transpositions); search does not notice this
 - pro: can be very memory-efficient
 - con: much wasted work (often exponentially slower)
- → first alternative usually preferable in planning (unlike many classical search benchmarks like 15-puzzle)

Searcr

Overview

Danneria





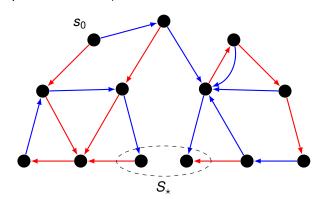
Example where search nodes correspond to operator sequences (no duplicate detection)



Progress

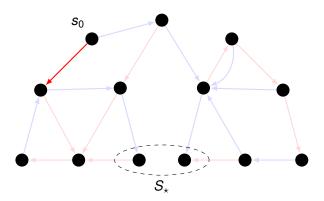
Overview

Regression





Example where search nodes correspond to operator sequences (no duplicate detection)



Search

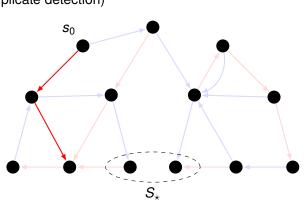
Progress

Example

Regression



Example where search nodes correspond to operator sequences (no duplicate detection)



Search

Progress

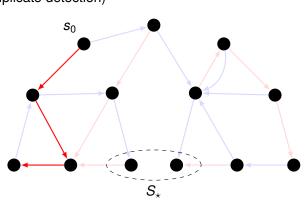
Overview

Regression





Example where search nodes correspond to operator sequences (no duplicate detection)



Search

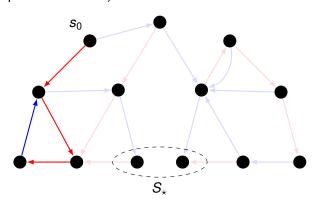
Progress

Example

Regression

UNI FREIB

Example where search nodes correspond to operator sequences (no duplicate detection)



Search

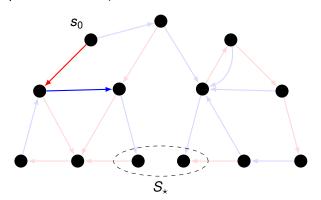
Progress

Example

Regression



Example where search nodes correspond to operator sequences (no duplicate detection)



Search

Progress

Example

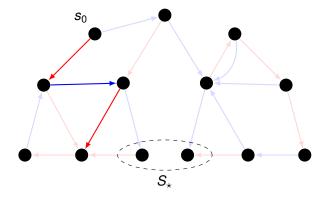
Regression



Example where search nodes correspond to operator sequences (no duplicate detection)



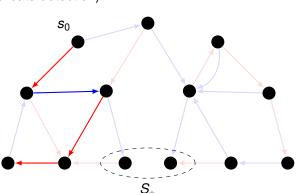
Regressio





Z W

Example where search nodes correspond to operator sequences (no duplicate detection)



Search

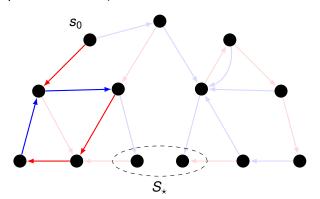
Progress

Overview

Regression



Example where search nodes correspond to operator sequences (no duplicate detection)



Search

Progress

Overview

Regression



Example where search nodes correspond to operator sequences (no duplicate detection)

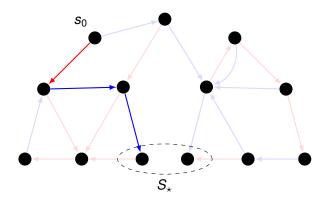


Ocarcii

Progress

Overview

Regression





Search

Progression

Regression

Overview Example

General case

Summary

Regression

Forward search vs. backward search



Going through a transition graph in forward and backward directions is not symmetric:

- forward search starts from a single initial state; backward search starts from a set of goal states
- when applying an operator o in a state s in forward direction, there is a unique successor state s'; if we applied operator o to end up in state s', there can be several possible predecessor states s

→ most natural representation for backward search in planning associates sets of states with search nodes

Search

Progression

Overview

Example

STRIPS

Practical issue

Summarv

Planning by backward search: regression



Regression: Computing the possible predecessor states $regr_o(G)$ of a set of states G with respect to the last operator o that was applied.

Regression planners find solutions by backward search:

- start from set of goal states
- iteratively pick a previously generated state set and regress it through an operator, generating a new state set
- solution found when a generated state set includes the initial state

Pro: can handle many states simultaneously
Con: basic operations complicated and expensive

Search

Progression

Overview

Example

STRIPS

Practical issues

Search space representation in regression planners





Search

Progression

Regression

Overview

STRIPS

General case

Summary

identify state sets with logical formulae (again):

- search nodes correspond to state sets
- each state set is represented by a logical formula: φ represents $\{s \in S \mid s \models \varphi\}$
- many basic search operations like detecting duplicates are NP-hard or coNP-hard





Search

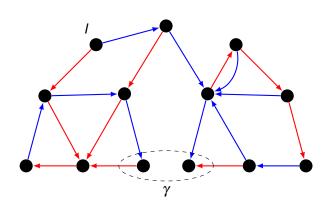
Progression

Regression

Overview Example

STRIPS

General case





AN E

γ

Search

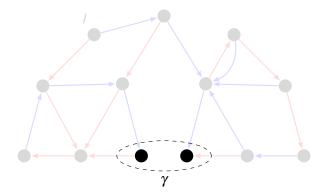
Progression

Regression

Overview Example

STRIPS General case

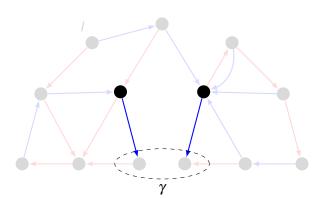
Practical issue





$$\varphi_1 = regr_{\longrightarrow}(\gamma)$$

$$\varphi_1 \longrightarrow \gamma$$



Search

Progression

Regression

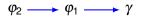
Overview Example

STRIPS

General case Practical issue



$$\varphi_1 = regr_{\longrightarrow}(\gamma)$$
 $\varphi_2 = regr_{\longrightarrow}(\varphi_1)$



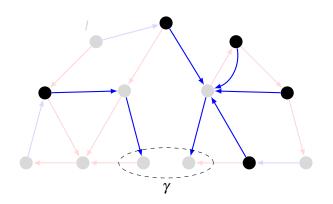


Progression

Regression Overview

Example

STRIPS General case





$$\varphi_1 = regr_{\longrightarrow}(\gamma)$$
 $\varphi_2 = regr_{\longrightarrow}(\varphi_1)$

$$\varphi_3 \longrightarrow \varphi_2 \longrightarrow \varphi_1 \longrightarrow$$

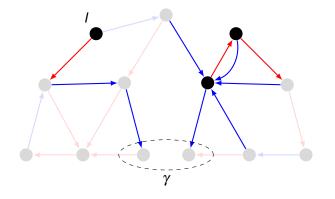
$$\varphi_3 = regr_{\longrightarrow}(\varphi_2), l \models \varphi_3$$



Regression

Overview Example

General case





Definition (STRIPS planning task)

A planning task is a STRIPS planning task if all operators are STRIPS operators and the goal is a conjunction of atoms.

Regression for STRIPS planning tasks is very simple:

- Goals are conjunctions of atoms $a_1 \wedge \cdots \wedge a_n$.
- First step: Choose an operator that makes none of a_1, \ldots, a_n false.
- Second step: Remove goal atoms achieved by the operator (if any) and add its preconditions.
- Outcome of regression is again conjunction of atoms.

Optimization: only consider operators making some a_i true

Search

Progression

_ .

Overview

General case

Summarv





Definition (STRIPS regression)

Let $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_n$ be a conjunction of atoms, and let $o = \langle \chi, e \rangle$ be a STRIPS operator which adds the atoms a_1, \ldots, a_k and deletes the atoms d_1, \ldots, d_l .

The STRIPS regression of φ with respect to o is

$$\mathit{sregr}_o(\varphi) := egin{cases} \bot & \text{if } a_i = d_j \text{ for some } i,j \ \bot & \text{if } \varphi_i = d_j \text{ for some } i,j \ \chi \land \bigwedge (\{\varphi_1, \ldots, \varphi_n\} \setminus \{a_1, \ldots, a_k\}) & \text{otherwise} \end{cases}$$

Note: $sregr_o(\varphi)$ is again a conjunction of atoms, or \bot .

STRIPS regression example





Progression

STRIPS

Summary









Note: Predecessor states are in general not unique. This picture is just for illustration purposes.

 $o_1 = \langle o_n \wedge c_{lr} \rangle$ $o_2 = \langle \bullet on \blacksquare \land \bullet clr \land \bullet clr, \neg \bullet clr \land \neg \bullet on \blacksquare \land \bullet on \blacksquare \land \bullet clr \rangle$

$$\neg \blacksquare on \blacksquare \land \blacksquare onT \land \blacksquare cIr \rangle$$





$$o_3 = \langle \blacksquare onT \land \blacksquare clr \land \blacksquare clr, \neg \blacksquare clr \land \neg \blacksquare onT \land \blacksquare on \blacksquare \rangle$$

$$\gamma = \blacksquare on \blacksquare \land \blacksquare on \blacksquare$$

$$\varphi_1 = sregr_{o_3}(\gamma) = \blacksquare onT \land \blacksquare clr \land \blacksquare clr \land \blacksquare on\blacksquare$$

$$\varphi_2 = sregr_{o_2}(\varphi_1) =$$
 on \land clr \land on \land

$$\varphi_3 = \ \textit{sregr}_{o_1}^{\ \ \ }(\varphi_2) = {\color{red}\blacksquare \textit{on}} {\color{red}\blacksquare} \land {\color{red}\blacksquare \textit{clr}} \land {\color{red}\blacksquare \textit{on}} {\color{red}\blacksquare} \land {\color{red}\blacksquare \textit{on}} T$$

Regression for general planning tasks



- With disjunctions and conditional effects, things become more tricky. How to regress $a \lor (b \land c)$ with respect to $\langle q, d \rhd b \rangle$?
- The story about goals and subgoals and fulfilling subgoals, as in the STRIPS case, is no longer useful.
- We present a general method for doing regression for any formula and any operator.
- Now we extensively use the idea of representing sets of states as formulae.

Search

Progression

Regression

Overview Example

General case

Practical issue:

Summarv

Definition (effect precondition)

The effect precondition $EPC_{I}(e)$ for literal I and effect e is defined as follows:

$$EPC_{I}(I) = \top$$

 $EPC_{I}(I') = \bot \text{ if } I \neq I' \text{ (for literals } I')$
 $EPC_{I}(e_{1} \wedge \cdots \wedge e_{n}) = EPC_{I}(e_{1}) \vee \cdots \vee EPC_{I}(e_{n})$
 $EPC_{I}(\chi \rhd e) = EPC_{I}(e) \wedge \chi$

Intuition: $EPC_{l}(e)$ describes the situations in which effect e causes literal I to become true

Progression

General case

Effect precondition examples



Search

Progression

Regression

Overview Example

STRIPS
General case

Practical issues

Summary

Example

$$\begin{array}{rcl} \textit{EPC}_a(b \land c) & = & \bot \lor \bot \equiv \bot \\ \textit{EPC}_a(a \land (b \rhd a)) & = & \top \lor (\top \land b) \equiv \top \\ \textit{EPC}_a((c \rhd a) \land (b \rhd a)) & = & (\top \land c) \lor (\top \land b) \equiv c \lor b \end{array}$$

Lemma (A)

Let s be a state, I a literal and e an effect. Then $I \in [e]_s$ if and only if $s \models EPC_I(e)$.

Proof.

Induction on the structure of the effect e.

Base case 1, e = I: $I \in [I]_s = \{I\}$ by definition, and $s \models EPC_I(I) = \top$ by definition. Both sides of the equivalence are true.

Base case 2, e = l' for some literal $l' \neq l$: $l \notin [l']_s = \{l'\}$ by definition, and $s \not\models EPC_l(l') = \bot$ by definition. Both sides are false.

Search

Progression

Regression

Example STRIPS

General case

0

Lemma (A)

Let s be a state, I a literal and e an effect. Then $I \in [e]_s$ if and only if $s \models EPC_I(e)$.

Proof.

Induction on the structure of the effect *e*.

Base case 1, e = I: $I \in [I]_s = \{I\}$ by definition, and $s \models EPC_I(I) = \top$ by definition. Both sides of the equivalence are true.

Base case 2, e = l' for some literal $l' \neq l$: $l \notin [l']_s = \{l'\}$ by definition, and $s \not\models EPC_l(l') = \bot$ by definition. Both sides are false.

Search

Progression

Regression

Example STRIPS

General case

Lemma (A)

Let s be a state, I a literal and e an effect. Then $I \in [e]_s$ if and only if $s \models EPC_I(e)$.

Proof.

Induction on the structure of the effect e.

Base case 1, e = I: $I \in [I]_s = \{I\}$ by definition, and $s \models EPC_I(I) = \top$ by definition. Both sides of the equivalence are true.

Base case 2, e = l' for some literal $l' \neq l$: $l \notin [l']_s = \{l'\}$ by definition, and $s \not\models EPC_l(l') = \bot$ by definition. Both sides are false.

Search

Progression

Regressio Overview

STRIPS General case

Practical issues

```
Inductive case 1, e = e_1 \wedge \cdots \wedge e_n:
                                                                 (Def [e_1 \wedge \cdots \wedge e_n]_s)
 I \in [e]_s iff I \in [e_1]_s \cup \cdots \cup [e_n]_s
             iff l \in [e']_s for some e' \in \{e_1, \dots, e_n\}
             iff s \models EPC_l(e') for some e' \in \{e_1, \dots, e_n\}
                                                                                           (IH)
             iff s \models EPC_l(e_1) \lor \cdots \lor EPC_l(e_n)
             iff s \models EPC_l(e_1 \land \cdots \land e_n).
                                                                                 (Def EPC)
```

General case





```
Inductive case 1, e = e_1 \wedge \cdots \wedge e_n: I \in [e]_s iff I \in [e_1]_s \cup \cdots \cup [e_n]_s (Def [e_1 \wedge \cdots \wedge e_n]_s) iff I \in [e']_s for some e' \in \{e_1, \dots, e_n\} iff s \models EPC_I(e') for some e' \in \{e_1, \dots, e_n\} (IH) iff s \models EPC_I(e_1) \vee \cdots \vee EPC_I(e_n) iff s \models EPC_I(e_1 \wedge \cdots \wedge e_n). (Def EPC)
```

```
Inductive case 2, e = \chi \rhd e':
I \in [\chi \rhd e']_s \text{ iff } I \in [e']_s \text{ and } s \models \chi \qquad \qquad \text{(Def } [\chi \rhd e']_s)
\text{iff } s \models EPC_I(e') \text{ and } s \models \chi \qquad \qquad \text{(IH)}
\text{iff } s \models EPC_I(e') \land \chi
\text{iff } s \models EPC_I(\chi \rhd e'). \qquad \qquad \text{(Def } EPC)
```

Search

Progression

Regression

Example STRIPS

General case Practical issues

Effect preconditions: connection to normal form





Remark: EPC vs. effect normal form

Notice that in terms of $EPC_a(e)$, any operator $\langle \chi, e \rangle$ can be expressed in effect normal form as

$$\left\langle \chi, \bigwedge_{a \in A} ((EPC_a(e) \rhd a) \land (EPC_{\neg a}(e) \rhd \neg a)) \right\rangle$$

where *A* is the set of all state variables.

Search

Progression

Regression

Overview Example

General case

Regressing state variables



The formula $EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$ expresses the value of state variable $a \in A$ after applying o in terms of values of state variables before applying o.

Either:

- a became true, or
- a was true before and it did not become false.

Search

Progression

Regression

Example

STRIPS
General case

Practical issue

Regressing state variables: examples



Example

Let
$$e = (b \triangleright a) \land (c \triangleright \neg a) \land b \land \neg d$$
.

variable <i>x</i>	$EPC_{x}(e) \lor (x \land \neg EPC_{\neg x}(e))$
а	$b \lor (a \land \neg c)$
b	
С	$\perp \vee (c \wedge \neg \bot) \equiv c$
d	$\perp \vee (d \wedge \neg \top) \equiv \perp$

Search

Progression

Regression

Overview

General case





Lemma (B)

Let a be a state variable, $o = \langle \chi, e \rangle$ an operator, s a state, and $s' = app_o(s)$. Then $s \models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$ if and only if $s' \models a$.

Proof

(⇒): Assume $s \models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$. Do a case analysis on the two disjuncts.

- Assume that $s \models EPC_a(e)$. By Lemma A, we have $a \in [e]_s$ and hence $s' \models a$.
- Assume that $s \models a \land \neg EPC_{\neg a}(e)$. By Lemma A, we have $\neg a \notin [e]_s$. Hence a remains true in s'.

Search

Progression

Regression

Example STRIPS

General case Practical issue





Lemma (B)

Let a be a state variable, $o = \langle \chi, e \rangle$ an operator, s a state, and $s' = app_o(s)$. Then $s \models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$ if and only if $s' \models a$.

Proof.

(⇒): Assume $s \models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$. Do a case analysis on the two disjuncts.

- Assume that $s \models EPC_a(e)$. By Lemma A, we have $a \in [e]_s$ and hence $s' \models a$.
- 2 Assume that $s \models a \land \neg EPC_{\neg a}(e)$. By Lemma A, we have $\neg a \notin [e]_s$. Hence a remains true in s'.

Search

Progression

Overview

STRIPS
General case

Practical issue



Lemma (B)

Let a be a state variable, $o = \langle \chi, e \rangle$ an operator, s a state, and $s' = app_o(s)$. Then $s \models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$ if and only if $s' \models a$.

Proof.

(⇒): Assume $s \models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$. Do a case analysis on the two disjuncts.

- Assume that $s \models EPC_a(e)$. By Lemma A, we have $a \in [e]_s$ and hence $s' \models a$.
- 2 Assume that $s \models a \land \neg EPC_{\neg a}(e)$. By Lemma A, we have $\neg a \notin [e]_s$. Hence a remains true in s'.

Search

Progression

Overview

General case Practical issues



Lemma (B)

Let a be a state variable, $o = \langle \chi, e \rangle$ an operator, s a state, and $s' = app_o(s)$. Then $s \models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$ if and only if $s' \models a$.

Proof.

 (\Rightarrow) : Assume $s \models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$. Do a case analysis on the two disjuncts.

- Assume that $s \models EPC_a(e)$. By Lemma A, we have $a \in [e]_s$ and hence $s' \models a$.
- 2 Assume that $s \models a \land \neg EPC_{\neg a}(e)$. By Lemma A, we have $\neg a \notin [e]_s$. Hence a remains true in s'.

Search

Progression

General case



Proof (ctd.)

(\Leftarrow): We showed that if the formula is true in s, then a is true in s'. For the second part, we show that if the formula is false in s, then a is false in s'.

- So assume $s \not\models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$.
- Then $s \models \neg EPC_a(e) \land (\neg a \lor EPC_{\neg a}(e))$ (de Morgan)
- Case distinction: a is true or a is false in s
 - Assume that $s \models a$. Now $s \models EPC_{\neg a}(e)$ because $s \models \neg a \lor EPC_{\neg a}(e)$.
 - Hence by Lemma A $\neg a \in [e]_s$ and we get $s' \not\models a$.
 - Assume that $s \not\models a$. Because $s \models \neg EPC_a(e)$, by Lemma A we get $a \notin [e]_s$ and hence $s' \not\models a$.

Therefore in both cases $s' \not\models a$.

Search

Progression

Regressio

Example STRIPS

General case Practical issues



 (\Leftarrow) : We showed that if the formula is true in s, then a is true in s'. For the second part, we show that if the formula is false in s, then a is false in s'.

- So assume $s \not\models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$.

Search

Progression

(\Leftarrow): We showed that if the formula is true in s, then a is true in s'. For the second part, we show that if the formula is false in s, then a is false in s'.

- So assume $s \not\models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$.
- Then $s \models \neg EPC_a(e) \land (\neg a \lor EPC_{\neg a}(e))$ (de Morgan).
- Case distinction: a is true or a is false in s
 - Assume that $s \models a$. Now $s \models EPC_{\neg a}(e)$ because $s \models \neg a \lor EPC_{\neg a}(e)$.
 - Assume that $s \not\models a$. Because $s \models \neg EPC_a(e)$, by Lemm
 - we get $a \notin [e]_s$ and hence $s' \not\models a$.

Therefore in both cases $s' \not\models a$.

Search

Progression

Overview Example

General case Practical issues

(\Leftarrow): We showed that if the formula is true in s, then a is true in s'. For the second part, we show that if the formula is false in s, then a is false in s'.

- So assume $s \not\models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$.
- Then $s \models \neg EPC_a(e) \land (\neg a \lor EPC_{\neg a}(e))$ (de Morgan).
- Case distinction: a is true or a is false in s.
 - Assume that $s \models a$. Now $s \models EPC_{\neg a}(e)$ because $s \models \neg a \lor EPC_{\neg a}(e)$.
 - 2 Assume that $s \not\models a$. Because $s \models \neg EPC_a(e)$, by Lemma A we get $a \notin [e]_s$ and hence $s' \not\models a$.

Therefore in both cases $s' \not\models a$.

Searci

Progression

Overview Example

General case

 (\Leftarrow) : We showed that if the formula is true in s, then a is true in s'. For the second part, we show that if the formula is false in s, then a is false in s'.

- So assume $s \not\models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$.
- Then $s \models \neg EPC_a(e) \land (\neg a \lor EPC_{\neg a}(e))$ (de Morgan).
- Case distinction: a is true or a is false in s.
 - 1 Assume that $s \models a$. Now $s \models EPC_{\neg a}(e)$ because $s \models \neg a \lor EPC_{\neg a}(e)$. Hence by Lemma A $\neg a \in [e]_s$ and we get $s' \not\models a$.

Progression

 (\Leftarrow) : We showed that if the formula is true in s, then a is true in s'. For the second part, we show that if the formula is false in s, then a is false in s'.

- So assume $s \not\models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$.
- Then $s \models \neg EPC_a(e) \land (\neg a \lor EPC_{\neg a}(e))$ (de Morgan).
- Case distinction: a is true or a is false in s.
 - 11 Assume that $s \models a$. Now $s \models EPC_{\neg a}(e)$ because $s \models \neg a \lor EPC_{\neg a}(e)$. Hence by Lemma A $\neg a \in [e]_s$ and we get $s' \not\models a$.
 - Assume that $s \not\models a$. Because $s \models \neg EPC_a(e)$, by Lemma A we get $a \notin [e]_s$ and hence $s' \not\models a$.

Therefore in both cases $s' \not\models a$.

Searc

Progression

Overview Example STRIPS

General case Practical issues

(\Leftarrow): We showed that if the formula is true in s, then a is true in s'. For the second part, we show that if the formula is false in s, then a is false in s'.

- So assume $s \not\models EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$.
- Then $s \models \neg EPC_a(e) \land (\neg a \lor EPC_{\neg a}(e))$ (de Morgan).
- Case distinction: a is true or a is false in s.
 - 1 Assume that $s \models a$. Now $s \models EPC_{\neg a}(e)$ because $s \models \neg a \lor EPC_{\neg a}(e)$. Hence by Lemma A $\neg a \in [e]_s$ and we get $s' \not\models a$.
 - 2 Assume that $s \not\models a$. Because $s \models \neg EPC_a(e)$, by Lemma A we get $a \notin [e]_s$ and hence $s' \not\models a$.

Therefore in both cases $s' \not\models a$.

Search

Progression

Overview Example STRIPS

General case Practical issues

Regression: general definition



We base the definition of regression on formulae $EPC_{l}(e)$.

Definition (general regression)

Let φ be a propositional formula and $o = \langle \chi, e \rangle$ an operator. The regression of φ with respect to o is

$$regr_o(\varphi) = \chi \wedge \varphi_r \wedge \kappa$$

where

■ φ_r is obtained from φ by replacing each $a \in A$ by $EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$, and

The formula κ expresses that operators are only applicable in states where their change sets are consistent.

Search

Progression

Overview Example

General case Practical issues

Regression examples



■
$$regr_{\langle a,b\rangle}(b) \equiv a \land (\top \lor (b \land \neg \bot)) \land \top \equiv a$$

■
$$regr_{\langle a,b\rangle}(b \land c \land d)$$

≡ $a \land (\top \lor (b \land \neg \bot)) \land (\bot \lor (c \land \neg \bot)) \land (\bot \lor (d \land \neg \bot)) \land \top$
≡ $a \land c \land d$

$$regr_{\langle a,c\rhd b\rangle}(b)\equiv a\wedge (c\vee (b\wedge\neg\bot))\wedge\top\equiv a\wedge (c\vee b)$$

■
$$regr_{\langle a,(c \rhd b) \land (b \rhd \neg b) \rangle}(b) \equiv a \land (c \lor (b \land \neg b)) \land \neg (c \land b)$$

≡ $a \land c \land \neg b$

■
$$regr_{\langle a,(c \rhd b) \land (d \rhd \neg b) \rangle}(b) \equiv a \land (c \lor (b \land \neg d)) \land \neg (c \land d)$$

 $\equiv a \land (c \lor b) \land (c \lor \neg d) \land (\neg c \lor \neg d)$
 $\equiv a \land (c \lor b) \land \neg d$

Search

Progression

Regression

Overview Example

General case

Summarv

Regression example: binary counter



$$\begin{array}{c} (\neg b_0 \rhd b_0) \land \\ ((\neg b_1 \land b_0) \rhd (b_1 \land \neg b_0)) \land \\ ((\neg b_2 \land b_1 \land b_0) \rhd (b_2 \land \neg b_1 \land \neg b_0)) \end{array}$$

$$\begin{split} EPC_{b_2}(e) &= \neg b_2 \wedge b_1 \wedge b_0 \\ EPC_{b_1}(e) &= \neg b_1 \wedge b_0 \\ EPC_{b_0}(e) &= \neg b_0 \\ EPC_{\neg b_2}(e) &= \bot \\ EPC_{\neg b_1}(e) &= \neg b_2 \wedge b_1 \wedge b_0 \\ EPC_{\neg b_0}(e) &= (\neg b_1 \wedge b_0) \vee (\neg b_2 \wedge b_1 \wedge b_0) \equiv (\neg b_1 \vee \neg b_2) \wedge b_0 \end{split}$$

Regression replaces state variables as follows:

$$\begin{array}{lll} b_2 & \text{by} & (\neg b_2 \wedge b_1 \wedge b_0) \vee (b_2 \wedge \neg \bot) \equiv (b_1 \wedge b_0) \vee b_2 \\ b_1 & \text{by} & (\neg b_1 \wedge b_0) \vee (b_1 \wedge \neg (\neg b_2 \wedge b_1 \wedge b_0)) \\ & & \equiv (\neg b_1 \wedge b_0) \vee (b_1 \wedge (b_2 \vee \neg b_0)) \\ b_0 & \text{by} & \neg b_0 \vee (b_0 \wedge \neg ((\neg b_1 \vee \neg b_2) \wedge b_0)) \equiv \neg b_0 \vee (b_1 \wedge b_2) \end{array}$$

Search

General case



Theorem (correctness of $regr_o(\varphi)$)

Let φ be a formula, o an operator and s a state. Then $s \models regr_o(\varphi)$ iff o is applicable in s and $app_o(s) \models \varphi$.

Proof.

Let $o = \langle \chi, e \rangle$. Recall that $regr_o(\varphi) = \chi \wedge \varphi_r \wedge \kappa$, where φ_r and κ are as defined previously.

If o is inapplicable in s, then $s \not\models \chi \land \kappa$, both sides of the "iff" condition are false, and we are done. Hence, we only further consider states s where o is applicable. Let $s' := app_o(s)$.

We know that $s \models \chi \land \kappa$ (because o is applicable), so the "iff" condition we need to prove simplifies to:

$$s \models \varphi_{\mathsf{r}} \text{ iff } s' \models \varphi.$$

Search

Progression

Overview

STRIPS
General case

Practical issues



FRE

Theorem (correctness of $regr_o(\varphi)$)

Let φ be a formula, o an operator and s a state. Then $s \models regr_o(\varphi)$ iff o is applicable in s and $app_o(s) \models \varphi$.

Proof.

Let $o = \langle \chi, e \rangle$. Recall that $regr_o(\varphi) = \chi \wedge \varphi_r \wedge \kappa$, where φ_r and κ are as defined previously.

If o is inapplicable in s, then $s \not\models \chi \land \kappa$, both sides of the "iff" condition are false, and we are done. Hence, we only further consider states s where o is applicable. Let $s' := app_o(s)$.

We know that $s \models \chi \land \kappa$ (because o is applicable), so the "iff" condition we need to prove simplifies to:

$$s \models \varphi_{\mathsf{r}} \text{ iff } s' \models \varphi.$$

Search

Progression

Dograccion

Overview Example

General case



FR

Progression

General case

Summary

Theorem (correctness of $regr_o(\varphi)$)

Let φ be a formula, o an operator and s a state. Then $s \models regr_o(\varphi)$ iff o is applicable in s and $app_o(s) \models \varphi$.

Proof.

Let $o = \langle \chi, e \rangle$. Recall that $regr_o(\varphi) = \chi \wedge \varphi_r \wedge \kappa$, where φ_r and κ are as defined previously.

If o is inapplicable in s, then $s \not\models \chi \land \kappa$, both sides of the "iff" condition are false, and we are done. Hence, we only further consider states s where o is applicable. Let $s' := app_o(s)$.

We know that $s \models \chi \land \kappa$ (because o is applicable), so the "iff" condition we need to prove simplifies to:

 $s\models arphi_{\mathsf{r}}$ iff $s'\models arphi$.



E E

Theorem (correctness of $regr_o(\varphi)$)

Let φ be a formula, o an operator and s a state. Then $s \models regr_o(\varphi)$ iff o is applicable in s and $app_o(s) \models \varphi$.

Proof.

Let $o=\langle \chi,e \rangle$. Recall that $regr_o(\varphi)=\chi \wedge \varphi_r \wedge \kappa$, where φ_r and κ are as defined previously.

If o is inapplicable in s, then $s \not\models \chi \land \kappa$, both sides of the "iff" condition are false, and we are done. Hence, we only further consider states s where o is applicable. Let $s' := app_o(s)$.

We know that $s \models \chi \land \kappa$ (because o is applicable), so the "iff" condition we need to prove simplifies to:

$$s \models \varphi_{\mathsf{r}} \text{ iff } s' \models \varphi.$$

Search

Progression

Overview Example

General case

To show: $s \models \varphi_r$ iff $s' \models \varphi$.

We show that for all formulae ψ , $s \models \psi_r$ iff $s' \models \psi$, where ψ_r is ψ with every $a \in A$ replaced by $EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$.

The proof is by structural induction on ψ

Induction hypothesis $s \models \psi_{\mathsf{r}}$ if and only if $s' \models \psi$.

Base cases 1 & 2 $\psi = \top$ or $\psi = \bot$: trivial, as $\psi_r = \psi$

Base case 3 $\psi = a$ for some $a \in A$: Then $\psi_r = EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$ Search

Progression

Dogranaion

Overview Example

General case

FIRCUCAL ISSUE





To show: $s \models \varphi_r$ iff $s' \models \varphi$.

We show that for all formulae ψ , $s \models \psi_r$ iff $s' \models \psi$, where ψ_r is ψ with every $a \in A$ replaced by $EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$.

The proof is by structural induction on ψ .

```
Induction hypothesis s \models \psi_r if and only if s' \models \psi.

Base cases 1 & 2 \psi = \top or \psi = \bot: trivial, as \psi_r = \psi.

Base case 3 \psi = a for some a \in A:

Then \psi_r = EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))

By Lemma B. s \models \psi_r iff s' \models \psi.
```

Search

Progression

Overview

STRIPS
General case

Practical issues





To show: $s \models \varphi_r$ iff $s' \models \varphi$.

We show that for all formulae ψ , $s \models \psi_r$ iff $s' \models \psi$, where ψ_r is ψ with every $a \in A$ replaced by $EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$.

The proof is by structural induction on ψ .

```
Induction hypothesis s \models \psi_r if and only if s' \models \psi.
```

Base case 3 $\psi = a$ for some $a \in A$: Then $\psi_r = EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$

By Lemma B, $s \models \psi_r$ iff $s' \models \psi$.

Search

Progression

Overview

STRIPS General case

Practical issues

To show: $s \models \varphi_r$ iff $s' \models \varphi$.

We show that for all formulae ψ , $s \models \psi_r$ iff $s' \models \psi$, where ψ_r is ψ with every $a \in A$ replaced by $EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$.

The proof is by structural induction on ψ .

Induction hypothesis $s \models \psi_r$ if and only if $s' \models \psi$.

Base cases 1 & 2 $\psi = \top$ or $\psi = \bot$: trivial, as $\psi_r = \psi$.

Search

Progression

General case





To show: $s \models \varphi_r$ iff $s' \models \varphi$.

We show that for all formulae ψ , $s \models \psi_r$ iff $s' \models \psi$, where ψ_r is ψ with every $a \in A$ replaced by $EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$.

The proof is by structural induction on ψ .

Induction hypothesis $s \models \psi_r$ if and only if $s' \models \psi$.

Base cases 1 & 2 $\psi = \top$ or $\psi = \bot$: trivial, as $\psi_r = \psi$.

Base case 3 $\psi = a$ for some $a \in A$: Then $\psi_r = EPC_a(e) \lor (a \land \neg EPC_{\neg a}(e))$. By Lemma B, $s \models \psi_r$ iff $s' \models \psi$. Search

Progression

Overview Example

General case Practical issues



NE NE

Proof (ctd.)

Inductive case 1 $\psi = \neg \psi'$:

$$\begin{split} s \models \psi_{\text{r}} \text{ iff } s \models (\neg \psi')_{\text{r}} \text{ iff } s \models \neg (\psi'_{\text{r}}) \text{ iff } s \not\models \psi'_{\text{r}} \\ \text{ iff } (\stackrel{\text{IH}}{}) \ s' \not\models \psi' \text{ iff } s' \models \neg \psi' \text{ iff } s' \models \psi \end{split}$$

Inductive case 2 $\psi = \psi' \lor \psi''$:

$$s \models \psi_{r} \text{ iff } s \models (\psi' \lor \psi'')_{r} \text{ iff } s \models \psi'_{r} \lor \psi''_{r}$$

$$\text{iff } s \models \psi'_{r} \text{ or } s \models \psi''_{r}$$

$$\text{iff (IH, twice) } s' \models \psi' \text{ or } s' \models \psi''$$

$$\text{iff } s' \models \psi' \lor \psi'' \text{ iff } s' \models \psi$$

Inductive case 3 $\psi = \psi' \wedge \psi''$: Very similar to inductive case 2, just with \wedge instead of \vee and "and" instead of "or".

Search

Progression

Regressio

Example

General case



Inductive case 1 $\psi = \neg \psi'$:

$$\begin{split} s \models \psi_{\mathsf{r}} \text{ iff } s \models (\neg \psi')_{\mathsf{r}} \text{ iff } s \models \neg (\psi'_{\mathsf{r}}) \text{ iff } s \not\models \psi'_{\mathsf{r}} \\ \text{ iff } (\mathsf{IH}) \ s' \not\models \psi' \text{ iff } s' \models \neg \psi' \text{ iff } s' \models \psi \end{split}$$

Inductive case 2 $\psi = \psi' \lor \psi''$:

$$s \models \psi_{r} \text{ iff } s \models (\psi' \lor \psi'')_{r} \text{ iff } s \models \psi'_{r} \lor \psi''_{r}$$

$$\text{iff } s \models \psi'_{r} \text{ or } s \models \psi''_{r}$$

$$\text{iff (IH, twice) } s' \models \psi' \text{ or } s' \models \psi''$$

$$\text{iff } s' \models \psi' \lor \psi'' \text{ iff } s' \models \psi$$

Inductive case 3 $\psi = \psi' \wedge \psi''$: Very similar to inductive case 2, just with \wedge instead of \vee and "and" instead of "or".

Progression

Overview

STRIPS
General case

Practical issue

Inductive case 1 $\psi = \neg \psi'$:

$$\begin{split} s \models \psi_{\mathsf{r}} \text{ iff } s \models (\neg \psi')_{\mathsf{r}} \text{ iff } s \models \neg (\psi'_{\mathsf{r}}) \text{ iff } s \not\models \psi'_{\mathsf{r}} \\ \text{ iff } (\mathop{\text{IH}}) \ s' \not\models \psi' \text{ iff } s' \models \neg \psi' \text{ iff } s' \models \psi \end{split}$$

Inductive case 2 $\psi = \psi' \lor \psi''$:

$$s \models \psi_{r} \text{ iff } s \models (\psi' \lor \psi'')_{r} \text{ iff } s \models \psi'_{r} \lor \psi''_{r}$$

$$\text{iff } s \models \psi'_{r} \text{ or } s \models \psi''_{r}$$

$$\text{iff (IH, twice) } s' \models \psi' \text{ or } s' \models \psi''$$

$$\text{iff } s' \models \psi' \lor \psi'' \text{ iff } s' \models \psi$$

Inductive case 3 $\psi = \psi' \wedge \psi''$: Very similar to inductive case 2, just with \wedge instead of \vee and "and" instead of "or".

Progression

Overview Example

General case

Emptiness and subsumption testing



The following two tests are useful when performing regression searches to avoid exploring unpromising branches:

- Test that $regr_o(\varphi)$ does not represent the empty set (which would mean that search is in a dead end). For example, $regr_{(a,\neg \rho)}(p) \equiv a \land \bot \equiv \bot$.
- Test that $regr_o(\varphi)$ does not represent a subset of φ (which would make the problem harder than before). For example, $regr_{\langle b,c \rangle}(a) \equiv a \wedge b$.

Both of these problems are NP-hard.

Searc

Progression

Regression

Overview Example STRIPS

Practical issues

Formula growth



The formula $regr_{o_1}(regr_{o_2}(\dots regr_{o_{n-1}}(regr_{o_n}(\varphi))))$ may have size $O(|\varphi||o_1||o_2|\dots|o_{n-1}||o_n|)$, i. e., the product of the sizes of φ and the operators.

 \rightsquigarrow worst-case exponential size $O(m^n)$

Logical simplifications

- $\blacksquare \ \bot \land \varphi \equiv \bot, \ \top \land \varphi \equiv \varphi, \ \bot \lor \varphi \equiv \varphi, \ \top \lor \varphi \equiv \top$
- $a \lor \varphi \equiv a \lor \varphi[\bot/a], \neg a \lor \varphi \equiv \neg a \lor \varphi[\top/a],$ $a \land \varphi \equiv a \land \varphi[\top/a], \neg a \land \varphi \equiv \neg a \land \varphi[\bot/a]$
- idempotency, absorption, commutativity, associativity, ...

Search

Progression

Regression

Example

General case

_

Restricting formula growth in search trees



Search

Progression

Regression

Example

STRIPS

Practical issues

Summary

Problem very big formulae obtained by regression

Cause disjunctivity in the (NNF) formulae (formulae without disjunctions easily convertible to small formulae $I_1 \wedge \cdots \wedge I_n$ where I_i are literals and n is at most the number of state variables.)

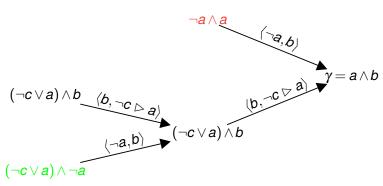
Idea handle disjunctivity when generating search trees

Unrestricted regression: search tree example



Unrestricted regression: do not treat disjunctions specially

Goal $\gamma = a \land b$, initial state $I = \{a \mapsto 0, b \mapsto 0, c \mapsto 0\}$.



Search

Progression

Pogrossion

Overview Example STRIPS

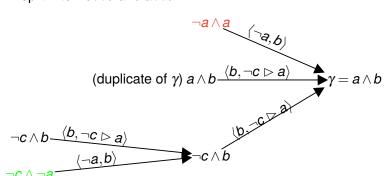
Practical issues

Full splitting: search tree example



Full splitting: always remove all disjunctivity

Goal
$$\gamma = a \wedge b$$
, initial state $I = \{a \mapsto 0, b \mapsto 0, c \mapsto 0\}$. $(\neg c \vee a) \wedge b$ in DNF: $(\neg c \wedge b) \vee (a \wedge b)$ \rightsquigarrow split into $\neg c \wedge b$ and $a \wedge b$



Search

Progression

Practical issues

General splitting strategies



Alternatives:

- Do nothing (unrestricted regression).
- Always eliminate all disjunctivity (full splitting).
- Reduce disjunctivity if formula becomes too big.

Discussion:

- With unrestricted regression the formulae may have size that is exponential in the number of state variables.
- With full splitting search tree can be exponentially bigger than without splitting.
- The third option lies between these two extremes.

Search

Progression

D-----

Example

STRIPS

Practical issues

- (Classical) search is a very important planning approach.
- Search-based planning algorithms differ along many dimensions, including
 - search direction (forward, backward)
 - what each search node represents (a state, a set of states, an operator sequence)
- Progression search proceeds forwards from the initial state.
 - If we use duplicate detection, each search node corresponds to a unique state.
 - If we do not use duplicate detection, each search node corresponds to a unique operator sequence.

Summary (ctd.)



Search

Flogression

Summary

ou.....a. y

- Regression search proceeds backwards from the goal.
 - Each search node corresponds to a set of states represented by a formula.
 - Regression is simple for STRIPS operators.
 - The theory for general regression is more complex.
 - When applying regression in practice, additional considerations such as when and how to perform splitting come into play.